Refine Search

Search Results -

Terms	Documents
(token$ and ((compar$ or match$) with (stor$ with hash$))) and @pd<=20001113	1

Database: US Pre-Grant Publication Full-Text Database; US Patents Full-Text Database; US OCR Full-Text Database; EPO Abstracts Database; JPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: L7

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Thursday, March 03, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,EPAB,JPAB,DWPI,TDBD; THES=ASSIGNEE; PLUR=YES; OP=OR			
L7	(token$ and ((compar$ or match$) with (stor$ with hash$))) and @pd<=20001113	1	L7
DB=USPT; THES=ASSIGNEE; PLUR=YES; OP=OR			
L6	L5 and 705/51,57,65.ccls.	3	L6
L5	(token$ and ((compar$ or match$) with (stor$ with hash$))) and @ad<=20001113	116	L5
L4	L3 and 705/51,57,65.ccls.	12	L4
L3	(token$ and ((compar$ or match$) with hash$)) and @ad<=20001113	381	L3
L2	L1 and (token$ or hash$)	1	L2
L1	5715403.pn.	1	L1

END OF SEARCH HISTORY

L6: Entry 1 of 3 File: USPT May 7, 2002

US-PAT-NO: 6385596
DOCUMENT-IDENTIFIER: US 6385596 B1

TITLE: Secure online music distribution system

DATE-ISSUED: May 7, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wiser; Philip R.	Redwood City	CA		
Cherenson; Andrew R.	Los Altos	CA		
Ansell; Steven T.	Fremont	CA		
Cannon; Susan A.	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Liquid Audio, Inc.	Redwood City	CA			02

APPL-NO: 09/ 020025 [PALM]
DATE FILED: February 6, 1998

INT-CL: [07] G06 F 17/60

US-CL-ISSUED: 705/51; 705/1, 705/57, 369/84, 380/201
US-CL-CURRENT: 705/51; 369/84, 380/201, 705/1, 705/57

FIELD-OF-SEARCH: 705/1, 705/24, 705/27, 705/26, 705/51, 705/52, 705/56, 705/57, 395/200.3, 395/200.31, 395/200.32, 395/200.33, 395/200.1, 395/200.06, 395/610, 380/3, 380/4, 380/21, 380/30, 380/5, 380/278, 380/282, 380/200, 380/201, 364/403, 364/479.04, 364/479.07, 369/84, 709/200, 709/201, 709/202, 709/203, 709/217, 700/234, 700/237

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4747139	May 1988	Taaffe	380/44
☐	4817140	March 1989	Chandra et al.	705/55
☐	5003410	March 1991	Endoh et al.	360/60
☐	5033084	July 1991	Beecher	705/56

☐	5034980	July 1991	Kubota	713/189
☐	5155768	October 1992	Matsuhara	463/29
☐	5191573	March 1993	Hair	369/84
☐	5199066	March 1993	Logan	713/200
☐	5418713	May 1995	Allen	364/403
☐	5592651	January 1997	Rackman	711/163
☐	5621796	April 1997	Davis et al.	380/24
☐	5623547	April 1997	Jones et al.	380/24
☐	5636276	June 1997	Brugger	380/4
☐	5661799	August 1997	Nagel et al.	705/52
☐	5673316	September 1997	Auerbach et al.	380/4
☐	5675734	October 1997	Hair	395/200.01
☐	5677953	October 1997	Dolphin	705/51
☐	5715314	February 1998	Payne et al.	705/78
☐	5724424	March 1998	Gifford	705/79
☐	5734823	March 1998	Saigh et al.	395/200.06
☐	5734891	March 1998	Saigh	395/610
☐	5742845	April 1998	Wagner	710/11
☐	5745568	April 1998	O'Connor et al.	705/56
☐	5754649	May 1998	Ryan et al.	380/203
☐	5757907	May 1998	Cooper et al.	705/52
☐	5778421	July 1998	Nagano et al.	711/115
☐	5790677	August 1998	Fox et al.	380/24
☐	5794217	August 1998	Allen	705/27
☐	5809144	September 1998	Sirbu et al.	705/53
☐	5857021	January 1999	Kataoka et al.	705/54
☐	5889860	March 1999	Eller et al.	380/4
☐	5892900	April 1999	Ginter et al.	713/200
☐	5900564	May 1999	Kurakake	84/477R
☐	5910987	June 1999	Ginter et al.	785/52
☐	5915019	June 1999	Ginter et al.	705/54
☐	5917912	June 1999	Ginter et al.	713/187
☐	5920861	July 1999	Hall et al.	707/9
☐	5943422	August 1999	Van Wiet et al.	705/54
☐	5949876	September 1999	Ginter et al.	705/80
☐	5982891	November 1999	Ginter et al.	705/54
☐	6005939	December 1999	Fortenberry et al.	705/76 X
	6061448	May 2000	Smith et al.	380/282

☐	6112181	August 2000	Shear et al.	705/1
☐	6138119	October 2000	Hall et al.	707/9
☐	6157721	December 2000	Shear et al.	380/255
☐	6185683	February 2001	Ginter et al.	713/176
☐	6189098	February 2001	Kaliski, Jr.	713/168
☐	6236971	May 2001	Stefik et al.	705/1
☐	6237786	May 2001	Ginter et al.	213/153
☐	6240185	May 2001	Van Wie et al.	380/232
☐	6253193	June 2001	Ginter et al.	705/57

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 309 298	March 1989	EP	
0 756 279	January 1997	EP	
WO 97/44736	November 1997	WO	

OTHER PUBLICATIONS

Atwood, Brent, "Liquid Audio gets Dolby license", Aug. 31, 1996; Billboard; New York; vol. 108, Issue 35, 3 pages.*
Blaze et al., "Decentralized Trust Management," originally published in Proc. IEEE Conference on Security and Privacy, Oakland, CA, May 1996.
Schneier and Kelsey, "Cryptographic Support for Secure Logs on Untrusted Machines," The Seventh USENIX Security Symposium Proceedings, USENIX Press, Jan. 1998, pp. 53-62.
Digital River, "Marketing Software on the Internet: A White Paper," Printed from website http://www.digitalriver.com/.
Digital River, "Technology Solutions to Electronic Transactions: A White Paper," Printed from website http://www.digitalriver.com/.
Digital River, "Fraud Prevention Technology," Printed from website http://www.digitalriver.com/.
One-page printed papers from website http://www.digitalriver.com/ on Digital River's "Mission," "Technology," "Security," "Privacy," and "How It Works".

ART-UNIT: 2131

PRIMARY-EXAMINER: Sough; Hyung-Sub

ATTY-AGENT-FIRM: Ivey; James D.

ABSTRACT:

A computer implemented online music distribution system provides for the secure delivery of audio data and related media, including text and images, over a public communications network. The online music distribution system provides security through multiple layers of encryption, and the cryptographic binding of purchased audio data to each specific purchaser. The online music distribution system also provides for previewing of audio data prior to purchase. In one embodiment, the

online music distribution system is a client-server system including a content manager, a delivery server, and an HTTP server, communicating with a client system including a Web browser and a media player. The content manager provides for management of media and audio content, and processing of purchase requests. The delivery server provides delivery of the purchased media data. The Web browser and HTTP server provide a communications interface over the public network between the content manager and media players. The media player provides for encryption of user personal information, and for decryption and playback of purchased media data. Security of purchased media data is enhanced in part by the use of a personal, digital passport in each media player. The digital passport contains identifying information that identifies the purchaser, along with confidential information, such as credit card number, and encryption data, such as the media player's public and private keys. The media player encryption data is used to encrypt purchased media data, which is decrypted in real time by the media player. The media player also displays confidential information, such as the purchaser's credit card number, during playback.

25 Claims, 29 Drawing figures

Previous Doc Next Doc Go to Doc#

L6: Entry 1 of 3 File: USPT May 7, 2002

DOCUMENT-IDENTIFIER: US 6385596 B1
TITLE: Secure online music distribution system

Application Filing Date (1):
19980206

Brief Summary Text (27):
The integrity of the purchase and delivery phases of a transaction are secured by a protocol between the content manager, delivery server, the user's Web browser, and media player that uses the purchaser's passport, and a separate trusted data object called a media voucher. The media voucher uniquely identifies the media being purchased, the specific purchase transaction, and the specific delivery server to deliver the purchased media to the media player. The specific purchase transaction is represented by a voucher ID generated by the content manager. The media voucher is provided by the content manager to the user's Web browser once the user's credit card has been checked and payment authorized. The content manager also provides a receipt token, a strong random number the media player will use to complete the transaction with the specified delivery server. This completes the purchase phase of the transaction.

Detailed Description Text (81):
The media player 116 then validates 622 the passport 400 for authentication and tamper detection by authenticating the certificate chain. The certificate chain is authenticated by starting with a root certificate of the media licensing center 110 that is stored in the media player 116, using the public key of the root certificate to decrypt a hash of the certificate and compare that decrypted hash with a newly generated hash. If the hashes are identical, the next certificate is authenticated in a same manner.

Detailed Description Text (118):
The voucher packet includes a voucher ID generated by the content manager 112 to track the reservation, a timestamp marking the start of the reservation, an expiration lifetime defining in seconds when the reservation becomes invalid after the timestamp, an authorization token that marks reservation as authorized, or as unauthorized in order to remove the reservation. Finally, the voucher packet includes a receipt token, which is returned in the media voucher to the media player 116 for initiating download of the requested media data file 200 from a delivery server 118. The authorization token is a secret token between the content manager 112 and the merchant server 132 and is not revealed to the user. This token and the receipt token are preferably strong random numbers.

Detailed Description Text (121):
ii) Purchased and not delivered. These are media data files 200 that have been authorized for delivery and for which a receipt token has been issued but not yet redeemed; and

Detailed Description Text (122):
iii) Purchased and delivered. These are media data files 200 for which a receipt token has been issued, validated, and redeemed by delivery of the file to the requesting media player 116.

Detailed Description Text (126):
Once the merchant server 132 has received payment authorization, it notifies the content manager 112 that the user has purchased the media associated with the voucher ID. This is done by providing 926 the voucher ID and authorization token previously sent to the merchant server 132, and a flag indicating the new state of the reservation as authorized for delivery. The content manager 112 updates the transaction database 130 to reflect that the voucher packet for this voucher ID has been authorized for purchase and download. This notification authorizes the content manager 112 to enable the requested media data file 200 for delivery. The content manager 112 returns 928 the voucher ID and an updated authorization token, which is needed in case the reservation needs modification.

Detailed Description Text (131):
where VVV is the voucher ID and RRR is the receipt token.

Detailed Description Text (132):
When a user clicks 934 on this link in the Web browser 128, another secure HTTP connection is setup by the Web browser 128 with the HTTP server 122, and the voucher ID and receipt token returned 936 to a CGI script that contacts 938 the content manager 112 to request the media voucher 300 containing the voucher ID, receipt token and delivery server network address and port number. The content manager 112 generates the media voucher 300 and returns 940 it to the Web browser 128 via the secure HTTP connection.

Detailed Description Text (135):
The media player 116 uses the receipt token (the shared secret with the content manager 112) to authenticate 946 the voucher ID 302 and the consumer certificate 402. The media player 116 establishes an unsecure TCP connection to the delivery server 118 using the address and port specified in the media voucher 300. The media player creates a message containing a keyed MAC of the voucher ID 302 using the receipt token as the key. This message is signed and sent 948 to the delivery server 118 to start the download procedure. The delivery server 118 sends 950 the encrypted data and the cleartext voucher ID 302 to the content manager 112 for verification.

Detailed Description Text (136):
The content manager 112 maps the voucher ID 302 to the receipt token in the transaction database 130. The content manager 112 then uses the receipt token to verify 952 the MAC encoded voucher ID and other data.

Detailed Description Text (138):
The content manager 112 then returns 956 the encrypted media key, along with audio quality information (bit rate and number of channels), the public key algorithm used with the media key itself and encryption parameters, the authorization token, media ID, the voucher ID, and the content manager's certificate serial number, and the media player's certificate number to the delivery server 118.

Detailed Description Text (140):
Once delivery is complete, the delivery server 118 notifies 962 the content manager 112, indicating the voucher ID, media ID, receipt token, time duration of the download, and the authorization token. The content manager 112 updates its transaction database 130 to reflect that the media data file has been delivered.

Detailed Description Text (166):
The commerce module 1012 also delivers media vouchers 300 to the media players 116, including the generation and validation of receipt tokens and authorization tokens.

Detailed Description Text (205):

Purchase Module 1106: This module manages a secure channel of communication based on a shared "secret" which is the receipt token that the security module 1004 generates as part of the media voucher 300. This module exports the following functions:

Current US Original Classification (1):
705/51

Current US Cross Reference Classification (4):
705/57

CLAIMS:

8. A computer-implemented method for distributing media data files including audio data to purchasers via a public communications network, the method comprising:

storing a plurality of media data files, each media data file including at least one audio image of a song encrypted with an associated media key, each media data file associated with a media ID for identifying the media data file;

receiving a request to purchase a selected one of the media data files, the request including the media ID of the selected media data file;

generating a voucher ID associated with the purchase of the selected media data file, a receipt token, and network addressing information of a delivery server to deliver the selected media data file;

responsive to receiving an authorization of the purchase of the selected media data file, transmitting the voucher ID, receipt token, and the network addressing information to a media player to receive the selected media data file;

receiving a data packet including an authenticated voucher ID including the receipt token, a public key of the media player, and a digital signature of the data packet formed using a private key of the media player;

responsive to successfully verifying the authenticated voucher ID against the firstmentioned voucher ID, encrypting the associated media key of the selected media data file with the public key of the media player to form an encrypted media key; and

authorizing delivery of the selected media data file by transmitting the encrypted media key and the media ID of the selected media data file.

22. A computer implemented online music distribution system for distributing digital media data files, including audio data, over a public communications network, the system comprising:

(a) a content manager that:

(i) receives a request to reserve a selected one of the media data files for a purchase transaction;

(ii) generates a media voucher including a voucher ID associated with the purchase transaction, a receipt token used to validate the voucher ID, a media ID identifying the selected media data file, and network addressing information of a delivery server to deliver the selected media data file;

(iii) transmits the media voucher to a client computer system including a media player for playing back the audio data of the selected media data file;

(b) the media player, storing a public key/private key pair assigned specifically to the media player, that:

(i) receives the media voucher from the content manager;

(ii) generates a data packet containing:

(1) data representing the voucher ID;

(2) a public key of the public key/private key pair of the media player; and

(3) a signature of the data packet formed using a private key of the public key/private key pair of the media player; and

(iii) transmits the data packet to the delivery server specified by the network addressing information in the media voucher; and

(c) the delivery server that:

(i) receives and parses the data packet, and transmits the voucher ID to the content manager;

(ii) receives from the content manager the selected media data file includes the audio data of the selected media data file encrypted with a media key, the media key encrypted with the public key of the media player; and

(iii) transmits the selected media data file to the media player, the media player adapted to playback the media data file by decryption of the media key with the private key.

Previous Doc Next Doc Go to Doc#

L6: Entry 3 of 3 File: USPT Apr 28, 1998

US-PAT-NO: 5745678
DOCUMENT-IDENTIFIER: US 5745678 A

TITLE: Method and system for the secured distribution of multimedia titles

DATE-ISSUED: April 28, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Herzberg; Amir	Bronx	NY		
Krawczyk; Hugo Mario	Bronx	NY		
Kutten; Shay	Rockaway	NJ		
Le; An Van	Sunnyvale	CA		
Matyas; Stephen Michael	Poughkeepsie	NY		
Yung; Marcel Mordechay	New York	NY		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 914911 [PALM]
DATE FILED: August 18, 1997

PARENT-CASE:
This is a continuation of application Ser. No. 08/354,700, filed Dec. 13, 1994, now abandoned.

INT-CL: [06] H04 L 9/00

US-CL-ISSUED: 395/186; 380/4
US-CL-CURRENT: 713/200; 705/51, 713/176

FIELD-OF-SEARCH: 395/186, 395/187.01, 395/188.01, 380/3, 380/4, 380/9, 380/23, 380/25

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL

☐	4295039	October 1981	Stuckert	235/380
☐	4309569	January 1982	Merkle	340/825.34
☐	4658093	April 1987	Hellman	380/25
☐	4789863	December 1988	Bush	340/825.34
☐	4908861	March 1990	Brachtl et al.	340/825.35
☐	5065429	November 1991	Lang	380/25
☐	5191613	March 1993	Graziano et al.	380/25
☐	5224166	June 1993	Hartman, Jr.	380/50
☐	5231666	July 1993	Matyas	380/25
☐	5241671	August 1993	Reed et al.	707/104
☐	5247575	September 1993	Sprague et al.	379/55.1
☐	5276738	January 1994	Hirsh	380/46
☐	5319705	June 1994	Halter	380/4
☐	5343527	August 1994	Moore	380/4
☐	5379343	January 1995	Grube et al.	380/4
☐	5421006	May 1995	Jablon et al.	380/4
☐	5432939	July 1995	Blackledge	395/700
☐	5450489	September 1995	Ostrover	380/3
☐	5485577	January 1996	Eyer	395/188.01
☐	5530751	June 1996	Morris	380/4
☐	5535188	July 1996	Dang	369/84
☐	5553139	September 1996	Ross	380/4
☐	5553143	September 1996	Ross	380/25

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
565 314 A3	October 1993	GB	
570 123 A1	November 1993	GB	

ART-UNIT: 243

PRIMARY-EXAMINER: Decady; Albert

ATTY-AGENT-FIRM: Salys; Casimer K. Venglarik; Daniel E. Dillon; Andrew J.

ABSTRACT:

A method and system for detecting authorized programs within a data processing system. The present invention creates a validation structure for validating a program. The validation structure is embedded in the program and in response to an initiation of the program, a determination is made as to whether the program is an

authorized program. The determination is made using the validation structure.

38 Claims, 8 Drawing figures

Previous Doc Next Doc Go to Doc#

L6: Entry 3 of 3 File: USPT Apr 28, 1998

DOCUMENT-IDENTIFIER: US 5745678 A
TITLE: Method and system for the secured distribution of multimedia titles

Application Filing Date (1):
19970818

Brief Summary Text (12):
Determining whether a multimedia program is an authorized multimedia program is accomplished by selecting a subset of the data objects within the multimedia program and validating the selected data objects using the validation structure stored in the multimedia program. This includes the steps of randomly selecting a portion of the data objects from among a defined set of data records listed in the validation structure, reading the selected data objects from the multimedia program using location information stored in the validation structure, and validating the selected data objects using validation information stored in the validation structure. For each selected data object, the location information stored in the validation structure is accessed and used to read the selected data object from the multimedia program. A cryptographic hash value is calculated on the selected data object and then compared for equality with a corresponding hash-value-of-reference stored in the validation structure The hash values must be equal for the selected data objects to be valid. In addition, the validation structure is itself validated through the use of the signature previously calculated on the validation structure, using a public key cryptographic algorithm, and stored within the validation structure. If the signature, validation structure, and subset of selected data objects are valid, the multimedia program is considered to be an authorized multimedia program. An authorized multimedia program is allowed to execute normally, otherwise, execution of the multimedia program may be prohibited or limited execution of the multimedia program may be allowed in response to a determination that the multimedia program is not an authorized program.

Drawing Description Text (7):
FIG. 5 is a block diagram of a signature token generation module, depicted in accordance with a preferred embodiment of the present invention;

Drawing Description Text (8):
FIG. 6 is a block diagram of a signature token validation module depicted in accordance with a preferred embodiment of the present invention;

Drawing Description Text (9):
FIG. 7 is a flowchart of a process for generating signature tokens in a signature token generation module depicted in accordance with a preferred embodiment of the present invention; and

Detailed Description Text (12):
Distribution of multimedia programs or titles (hereinafter called "multimedia titles") involves an application developer who produces multimedia titles using an authoring tool and a Run Time Environment (RTE) provided by a multimedia company and a user who purchases multimedia titles for execution on a computer or computer

platform executing the RTE. In accordance with a preferred embodiment of the present invention, checking for authorized multimedia titles and detecting unauthorized multimedia titles involves a scheme of digital signatures using a public key algorithm. A "public key" is a key made available to anyone who wants to encrypt information. In public key cryptography, public key algorithms are used in which a public key is used for encryption and a private key is used for decryption. The basis for public key cryptography includes discrete logarithms, factoring, and the knapsack problem. Each authorized multimedia title includes an embedded digital signature token that can be verified by the RTE before the multimedia title is permitted to execute on the data processing system.

Detailed Description Text (13):
Two cryptographic subsystems are employed to facilitate the signature token generation and signature token verification processes in accordance with a preferred embodiment of the present invention. One cryptographic subsystem enables the generation of signature tokens that, when embedded in authorized multimedia titles, will permit these titles to be validated. Another cryptographic subsystem is employed to validate the signature tokens. In this manner authorized multimedia titles may be distinguished from unauthorized multimedia titles.

Detailed Description Text (14):
With reference to FIG. 3, a block diagram of a creation and distribution process for multimedia titles on CD-ROM is depicted in accordance with a preferred embodiment of the present invention. Those skilled in the art will recognize that the subject invention could be practiced in an implementation wherein multimedia titles are distributed on media other than a CD ROM medium. A multimedia title is developed by a developer using authoring tool 300. The multimedia title is then processed using signature token generation module 302. This module generates a signature token for the multimedia title. The signature token is embedded within the multimedia title. Thereafter, the multimedia title with the signature token embedded within it is sent back to the developer who creates a master CD-ROM 304. Alternatively, the signature token and multimedia title are sent back to the developer, whereupon the signature token is embedded into the multimedia title and a master CD-ROM 304 is created by the developer. From master CD-ROM 304, CD-ROM 306 is produced containing the multimedia title and the embedded signature token. CD-ROM 306 may be placed within data processing system 308, which includes the RTE with the signature token validation module in accordance with a preferred embodiment of the present invention. When the title is to be executed within data processing system 308, the RTE reads the signature token from the CD-ROM and validates the signature token and a selected portion of the data objects also read from the CD-ROM using the signature token validation module.

Detailed Description Text (21):
With reference to FIG. 5, a diagram of a signature token is depicted in accordance with a preferred embodiment of the present invention. Signature token 500 is constructed by a signature token generation module (not shown in FIG. 5). The signature token is constructed step-by-step by making repeated service requests to the signature token generation module. Once created, signature token 500 is embedded in the multimedia title upon which it was generated. This signature token is validated by a signature token validation module in the RTE. In validation, the signature token is validated step-by-step by making repeated service requests to the signature token validation module.

Detailed Description Text (22):
Signature token 500 includes a header 502 and data records 1 through n that correspond to data or data objects in the multimedia title that can be selected and validated. The data records 1 through n in the signature token are different from the data objects in the multimedia title, although there is a direct correspondence. In addition, signature token 500 includes digital signature 504, which is employed to validate the header and the series of data records 1 through n

in the signature token. Each data record within signature token 500 includes location specific information, L, and a cryptographic hash value, H. Location specific information tells the signature token validation module the location or locations in the multimedia title of the data to be validated. The hash value is calculated on the specific multimedia data referenced by L. In accordance with a preferred embodiment of the present invention, the cryptographic hash value is calculated using a one-way function. A one-way function is one where it is computationally infeasible to find two different inputs X and Y, such that the cryptographic hash of X is equal to the cryptographic hash of Y. The term "computationally infeasible" is one used in prior art to describe a mathematical procedure that cannot be performed in the practical sense because of the very large number of computational steps required. However, the term is not precise in that there is no prescribed number of computations above which a computation is said to be computationally infeasible and below which the computation is said to be computationally feasible. In general, a mathematical procedure is said to be computationally infeasible if the cost or time to perform the necessary computations is beyond reasonable human means, e.g., if all the computers in the world linked together couldn't solve the problem in a billion billion billion years.

Detailed Description Text (23):
Location specific information may be a combination of location specific information stored in signature token 500 and location specific information derived algorithmically. The table of contents in a multimedia title is one example of location specific information that can be omitted from signature token 500. In other words, signature token 500 does not need to store an address or location of the table of contents because a simple procedure exists for always finding the table of contents given a standard starting point.

Detailed Description Text (24):
A portion of the data records 1 through n in signature token 500 will reference data in one or more data objects in the multimedia title. The signature token generation module employs a process to select a subset of different data objects (referenced by the object-property pairs in the table of contents) from a multimedia title to be validated. This process will be described in more detail below. It is desirable to have an element of randomness in this process although strictly speaking random in this process is not required. Once the subset of data objects in the multimedia title have been identified, a subportion of the data in each data object is selected and the locations of these subportions of data (denoted L1, L2, . . . , Ln) are used by the signature token generation module to read the data associated with each subportion. The signature token generation module then issues a service request to the cryptographic subsystem to generate a cryptographic hash value Hi on each subportion of data referenced by location information Li. The cryptographic hash value, Hi, is then stored in data record i together with the location information, Li. After the header and data records 1 through n have been created the signature token generation module issues a service request to the cryptographic subsystem to calculate a cryptographic hash value on the signature token, except for the digital signature. The signature token generation module then issues a service request to the cryptographic subsystem to calculate a digital signature on the signature token, using the cryptographic hash value calculated on the signature token and the private key of the multimedia company. The digital signature is then stored in the signature token.

Detailed Description Text (25):
The signature token validation module will randomly select and validate a subset of the data records in the signature token thus introducing an element of randomness into the validation process in accordance with a preferred embodiment of the present invention. Once the subset of data records has been randomly selected, the signature token validation module will process each data record on a record-by-record basis. The location specific information, L, in the data record is used to

read the referenced data from the CD-ROM. A service request is then issued to the cryptographic subsystem to generate a cryptographic hash value, H, on the referenced data. This calculated cryptographic hash value is then compared for equality with the reference cryptographic hash value, H, stored in the data record. If the cryptographic hash values are equal, the process then continues. Otherwise, processing may be halted with an indication that validation has failed. After each data record is processed, the signature token validation module issues a service request to the cryptographic system to cryptographically hash the signature token (except for the digital signature) that it previously read from the multimedia title. Then a service request is issued to the cryptographic subsystem to validate the digital signature. In accordance with a preferred embodiment of the present invention, the digital signature is encrypted with the public key of the multimedia company, stored in the signature token validation module. The encrypted value of the digital signature contains a hash-value-of-reference previously calculated on the valid signature token. The calculated hash value is then compared for equality with the so-obtained hash-value-of-reference. If the hash values are equal, then the signature token and digital signature are valid. Otherwise, the signature token and digital signature are not valid. Those skilled in the art will recognize that the subject invention can be practiced using any digital signature method without departing from the spirit of the invention.

Detailed Description Text (26):
The digital signature also can be calculated on a cryptographic hash value representing the root of a tree of cryptographic hash values, e.g., a binary tree of cryptographic hash values as described in U.S. Pat. No. 4,309,569, "Method of Providing Digital Signatures". By storing n-1 additional intermediate cryptographic hash values in the signature token, m of the possible n data objects in the signature token can be validated using m*log2n hashing operations instead of n hashing operations, which for small n may be more advantageous. A method of calculating the tree of cryptographic hash values is described in U.S. Pat. No. 4,309,569 and in U.S. Pat. No. 5,231,666, "Cryptographic Method For Updating Financial Records".

Detailed Description Text (27):
With reference now to FIG. 6, a block diagram of a signature token generation module and a signature token validation module is depicted in accordance with a preferred embodiment of the present invention. Signature token generation module 600 includes generation module 602 and cryptographic subsystem 604. Signature validation module 606 includes a validation program 608 and a cryptographic subsystem 610. Generation program 602 in signature token generation module 600 selects data (including data in randomly selected data objects) in the multimedia title to be validated. Generation program 602 reads data from the multimedia title and processes the data by issuing repeated service requests to cryptographic subsystem 604. These repeated service requests to process data are used to build a signature token. Similarly, validation program 608 in signature token validation module 606 randomly selects (for subsequent processing and validation) a subset of the data records in the signature token generated by signature token generation module 600. This data is read from the CDROM and is processed by issuing repeated service requests to cryptographic subsystem 610 to validate the signature token.

Detailed Description Text (28):
Cryptographic subsystem 604 provides the following cryptographic services to generation program 602: (1) initialize random number generator, (2) generate random number, (3) generate hash value, (4) generate digital signature, and (5) verify digital signature. A verify digital signature function is provided so that once a signature token is generated, generation program 602 can validate the signature token to insure that the signature token can be correctly processed by cryptographic subsystem 610 and signature token module 606. Such a verification function provides a high integrity process in creating multimedia titles with embedded signature tokens. Cryptographic subsystem 610 in signature token

validation module 606 supports validation program 608 by providing the following services: (1) initialize random number generator, (2) generate random number, (3) generate hash value, and (4) verify digital signature. The random number generation is employed to randomly select data records in the signature token to be validated. Algorithms and procedures for generating random numbers, generating hash values, and for generating and verifying digital signatures are well know within the prior art.

Detailed Description Text (34):
With reference now to FIG. 7, a flowchart of a process for generating signature tokens in a signature token generation module is depicted in accordance with a preferred embodiment of the present invention. The process begins by receiving a multimedia title with the signature token (step 700). Thereafter, the process fixes variables N and M such that N is the maximum number of data records to be created for the signature token that is to be produced, and M is the maximum record data length in bytes of each selected data object (step 702). For example, N=1,000 and M=1,000 are possible values for N and M. The process then locates and reads the table of contents or equivalent information contained in the multimedia title (step 704). This information is called "D1". In accordance with a preferred embodiment of the present invention, a 128 bit MDC, designated MDC 1, is calculated on the data D1 (step 705). Using the table of contents as necessary, the first logo screen is located and read from the multimedia title (step 706). The logo screen contains the name of the multimedia title and is designated "D2". A 128 bit MDC, designated MDC 2, is calculated on the data D2 (step 707).

Detailed Description Text (37):
When X=R, the process then proceeds to build a signature token, such as signature token 500 depicted in FIG. 5, consisting of a header, n data records, and a digital signature (step 722). The token type (e.g., "multimedia signature token") and n (number of data records) is stored within the header. In the first data record, the data location and the hash value are set to the constant value "table of contents" and the calculated value of MDC 1, respectively. The term "table of contents" is stored in the data location sub-field in lieu of location information because the signature token validation module always knows the location of the table of contents in accordance with a preferred embodiment of the present invention. In the second data record in the signature token, the data location and hash value are set to the constant value "logo screen" and the calculated value of MDC2, respectively. The term "logo screen" is stored in the data location sub-field in lieu of location information because the signature token validation module is able to calculate the location of the logo screen from the information found in the table of contents in accordance with a preferred embodiment of the present invention. The data locations and hash values in data records 3, 4, . . . , n are set to (L3, MDC3), (L4, MDC4), . . . , (Ln, MDCn), respectively. Note also that the hash values H1 through Hn in data records 1 through n of the signature token are equal to the MDC values: MDC1 through MDCn, respectively.

Detailed Description Text (38):
Furthermore, the signature token also includes space for a digital signature in accordance with a preferred embodiment of the present invention. An RSA digital signature based on ISO Standard 9796 might typically require from 512 to 2048 bits of storage space. A DSS digital signature based on FIPS 186 requires 320 bits of storage space. At step 722, the digital signature portion of the signature token is presently uninitialized. Thereafter, the process calculates a digital signature for the signature token with the multimedia company's private key (step 724). More specifically, a 128 bit MDC is calculated on the portion of the signature token consisting of the header and the n data records. This calculation may be performed using the process described in U.S. Pat. No. 4,908,861 entitled "Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function".

Detailed Description Text (39):
Thereafter, a digital signature is calculated on the MDC value using the private key of the multimedia company in accordance with a preferred embodiment of the present invention. The digital signature could be calculated with an RSA private key in accordance with ISO standard 9796 or with a DSA private key in accordance with FIPS 186. Those skilled in the art will recognize that private keys of different lengths can be employed to calculate the digital signature, and that digital signatures of different lengths can be stored in the signature token in accordance with a preferred embodiment of the present invention. Thereafter, the digital signature is stored in the signature token and the process terminates.

Detailed Description Text (40):
With reference now to FIG. 8, a flowchart of a process for validating multimedia titles in a validation program is depicted in accordance with a preferred embodiment of the present invention. This process is employed within the signature token validation module 606 of FIG. 6. The process begins by receiving a multimedia title with the embedded signature token (step 800). The process then fixes variable R, where R in this flowchart is the number of data records to be randomly authenticated (step 802). For example, R may be any value, such as 3, 4, 5, etc., but will generally be a small value in order to minimize the required processing time. Thereafter, the process reads the embedded signature token (step 804). In accordance with a preferred embodiment of the present invention, the table of contents may be used to locate and read the embedded signature token.

Detailed Description Text (41):
The process then locates and reads the table of contents or equivalent information contained in the multimedia title (step 806). This information is called "D1". The process then calculates a 128 bit MDC on the table of contents, D1 (step 808). This calculation may be performed in various ways. For example, the process described in U.S. Pat. No. 4,908,861, entitled "Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function" may be used. The process then validates the MDC calculated on the table of contents against the MDC-of-reference stored in the signature token (step 810). The 128 bit MDC calculated in step 808 is compared with H1 in the first data record in the signature token. The process then determines whether the two values are equal (step 812). If MDC is not equal to H1, the process then indicates that the title is invalid (step 814). In accordance with a preferred embodiment of the present invention, an invalid multimedia title so-detected by the signature token validation module causes an indicator or indication to be set, but does not halt processing. However, the present invention could be practiced using an implementation wherein the processing is halted as soon as an invalid condition is detected, which would not depart from the spirit of the present invention.

Detailed Description Text (42):
If, however, MDC is equal to H1, the process then locates and reads the first logo screen that is displayed to a user in the multimedia title (step 816). The table of contents may be used to locate and read this logo screen. The logo screen contains the name of the multimedia title, and this information is called "D2". Next, an MDC is calculated on the data comprising the logo screen (step 818). A 128 bit MDC is calculated in accordance with a preferred embodiment of the present invention. The process then validates the MDC calculated on the logo screen against the MDC stored in the signature token (step 820). The 128 bit MDC calculated in step 818 is compared with H2 in the second data record in the signature token. The process then determines whether the two values are equal (step 822). If MDC is not equal to H2, the process then indicates that the title is invalid (step 824). The process then sets Y=1 (step 826).

Detailed Description Text (43):
The process also sets Y=1 if MDC is equal H2 (step 826). The process then randomly selects and reads one of the n-2 remaining data records in the signature token

(step 828). The data records are selected from data record 3 through data record n, inclusively. Each data record is associated with and refers to all or a portion of the data for a single object-property pair in the multimedia title. The process then reads the data OP1 associated with the object-property pair pointed to by the selected data record (step 830). The data location information in the data record is used to locate and read the data OP1 associated with the referenced object-property pair in the multimedia title.

Detailed Description Text (44):
Next, the process calculates a 128 bit MDC on the selected data OP1 (step 832). The process then validates the MDC calculated on the data OP1 against the corresponding MDC, or hash value H, stored in the data record read from the signature token (step 834). In step 834 the 128 bit MDC calculated in step 832 is compared with the corresponding MDC, or hash value H, recovered from the data record. If the calculated MDC and stored MDC are equal, control is passed to step 838. If the MDC values are not equal, the process indicates that the title is invalid (step 836) and control is passed to (step 838). The process determines whether Y=R (step 838). If Y is not equal to R, the process sets Y=Y+1 (step 839) and returns to step 828. This continues until R data records have been randomly selected from among data records 3 through n in the signature token, and processed. When Y=R, the process then proceeds to validate the digital signature and signature token. Next, the process reads the digital signature in the signature token (step 840).

Detailed Description Text (45):
Thereafter, the process accesses the multimedia company's public key, which is stored as a fixed constant in the signature token validation module (step 842). The process then validates the signature using the public key (step 844). A 128 bit MDC is calculated on the portion of the signature token consisting of the header and the n data records. When an RSA digital signature is employed, the original 128 bit hash value H is recovered from the RSA digital signature using the process described in ISO standard 9796 utilizing the RSA public key. Thereafter, the 128 bit MDC calculated on the signature token including the header and the n data records are compared with the 128 bit hash value H. If these MDCs are equal, then the multimedia title is accepted as valid. Otherwise, the process indicates that the title is invalid. When a digital signature based on FIPS 186 is employed, the signature is validated by following the steps outlined in FIPS 186, which are different from those used to validate an RSA digital signature.

Current US Cross Reference Classification (1):
705/51

CLAIMS:

35. A storage device readable by a data processing system and encoding data processing system executable instructions for validating a program, wherein the program includes a validation structure having a plurality of data records, wherein each data record within the plurality of data records includes a cryptographic hash value for program data within a section other than a lead-in section of the program and a location value, wherein the location value indicates a location of the section, the storage device comprising:

creation means for creating a cryptographic hash value on program data within the section in the location indicated by the location value for the randomly selected data record for each randomly selected data record; and

comparison means for comparing the created cryptographic hash value with the hash value within the randomly selected data record, wherein the means are activated when the storage device is connected to and accessed by a data processing system.

L7: Entry 1 of 1 File: JPAB Jan 17, 1991

PUB-NO: JP403010387A
DOCUMENT-IDENTIFIER: JP 03010387 A
TITLE: COLOR EXTENDING SYSTEM FOR DATA DRIVING TYPE PROCESSOR

PUBN-DATE: January 17, 1991

INVENTOR-INFORMATION:
NAME COUNTRY
NAITOU, HIROMIKI

ASSIGNEE-INFORMATION:
NAME COUNTRY
SHARP CORP

APPL-NO: JP01145999
APPL-DATE: June 7, 1989

INT-CL (IPC): G06F 15/82

ABSTRACT:

PURPOSE: To improve the flexibility at the time of generating an application program and to quickly execute the data processing by executing an execution sequence management at the time of hash collision by comparing whether a real color and a real generation code are large or small.

CONSTITUTION: A hash collision processing part 20 in a waiting part 2 refers to a real color storage part 19, based on a temporary color code of tokens which come into collision with each other and compares mutually whether a real color code or a executing generation code corresponding to its temporary color code is large or small. Subsequently, the management of an execution sequence is executed by allowing the token of a large real color code to hold preferentially in a working memory or sending out preferentially the token of a large real generation code to a turn-round pipeline 12. Accordingly, at the time of executing a color obtaining instruction, free temporary colors are all utilized, and the real color code of 32 bits can be utilized freely extending over the full width for the execution sequence management. In such a way, the flexibility at the time of generating an application program is improved remarkably, and also, the data processing can be quickened remarkably.

L6: Entry 2 of 3 File: USPT Jul 17, 2001

US-PAT-NO: 6263446
DOCUMENT-IDENTIFIER: US 6263446 B1
**** See image for Certificate of Correction ****

TITLE: Method and apparatus for secure distribution of authentication credentials to roaming users

DATE-ISSUED: July 17, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kausik; Balas Natarajan	Los Gatos	CA		
Varadarajan; Rammohan	Cupertino	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Arcot Systems, Inc.	Santa Clara	CA			02

APPL-NO: 09/ 196430 [PALM]
DATE FILED: November 19, 1998

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATIONS This application is a Continuation-in-Part of pending U.S. patent application Ser. No. 08/996,758 filed Dec. 23, 1997.

INT-CL: [07] G06 F 11/30

US-CL-ISSUED: 713/201; 713/153, 713/155, 713/156, 713/159, 380/259, 380/264, 380/282, 380/286, 705/56, 705/64, 705/65, 705/66, 705/67, 705/73
US-CL-CURRENT: 713/201; 380/259, 380/264, 380/282, 380/286, 705/56, 705/64, 705/65, 705/66, 705/67, 705/73, 713/153, 713/155, 713/156, 713/159

FIELD-OF-SEARCH: 380/259, 380/264, 380/282, 380/286, 705/56, 705/64, 705/65, 705/66, 705/67, 705/71, 705/72, 705/73, 705/76, 713/153, 713/155, 713/156, 713/159, 713/201

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	3798605	March.1974	Feistel	340/172.5

☐	5148479	September 1992	Bird et al.	380/23
☐	5475756	December 1995	Merritt	380/24
☐	5491752	February 1996	Kaufman et al.	380/30
☐	5668876	September 1997	Falk et al.	380/25
☐	5689566	November 1997	Nguyen	380/25
☐	5757918	May 1998	Hopkins	380/25
☐	5764890	June 1998	Glasser et al.	395/188.01
☐	5778065	July 1998	Hauser et al.	380/21

ART-UNIT: 272

PRIMARY-EXAMINER: Swann; Tod

ASSISTANT-EXAMINER: Callahan; Paul E.

ATTY-AGENT-FIRM: Yang; Joseph Skadden, Arps, Slate, Meagher & Flom LLP

ABSTRACT:

A roaming user needing an his authentication credential (e.g., private key) to access a computer server to perform an electronic transaction may obtain the authentication credential in an on-demand fashion from a credential server accessible to the user over a computer network. In this way, the user is free to roam on the network without having to physically carry his authentication credential. Access to the credential may be protected by one or more challenge-response protocols involving simple shared secrets, shared secrets with one-to-one hashing, or biometric methods such as fingerprint recognition. If camouflaging is used to protect the authentication credential, decamouflaging may be performed either at the credential server or at the user's computer.

52 Claims, 3 Drawing figures

Previous Doc Next Doc Go to Doc#

L6: Entry 2 of 3 File: USPT Jul 17, 2001

DOCUMENT-IDENTIFIER: US 6263446 B1
** See image for Certificate of Correction **
TITLE: Method and apparatus for secure distribution of authentication credentials to roaming users

Application Filing Date (1):
19981119

Brief Summary Text (3):
To overcome the insecurity of the password, alternative technologies have been developed. One such technology is asymmetric key cryptography. In this technology, each user has two keys, a private key and a public key. The user performs a cryptographic operation (e.g., an encryption or a digital signature) on a digital quantity using his private key, such that the quantity may be authenticated by a verifier having access only to the user's public key. The private key therefore serves as the user's authentication credential. That is, the verifier need not know the user's private key in order to authenticate the user. Because the public key may be widely disseminated while the private key remains confidential, strong authentication is provided with enhanced security. Private keys are generally too long and complex for the user to memorize, and are therefore usually stored in software or hardware tokens, and interfaced with computers prior to use.

Brief Summary Text (4):
One such software token is the so-called software wallet, in which the private key is encrypted with a password or other access-controlled datum. In such software wallets, an intruder is not deterred from repeatedly trying passwords, in an exhaustive manner, until he recovers the private key. This poses analogous security risks to the simple password schemes described above. In addition, the software wallet is stored on a user's computer, which may be inconvenient if the user needs to freely roam from one location to another.

Brief Summary Text (5):
In contrast to software wallets, hardware tokens such as smart cards are more secure, and can be conveniently carried as the user roams. In a typical hardware smart card, the private key is stored in hardware, and protected by a watchdog chip that allows the user to access the private key, should he enter the correct password that unlocks the smart card. The smart card can even be configured so that, if a hacker attempts to guess passwords, the card locks up after a small number of successive missed attempts. The disadvantages of hardware token are: (1) roaming is restricted to locations where the appropriate token reader hardware is installed; (2) hardware tokens are expensive in contrast to software tokens; (3) hardware tokens must be physically carried wherever the user wishes to roam; and (4) hardware tokens are often lost, misplaced, or stolen.

Brief Summary Text (6):
Thus, while hardware token systems offer increased security, they have several disadvantages compared to software based systems. It would, therefore, be desirable to have a system that combines the best features of both hardware and software based systems.

Brief Summary Text (8):
The present invention discloses a method and apparatus for the on-demand delivery of authentication credentials to roaming users. Credentials are stored, delivered and transmitted in software, obviating the need for additional hardware. In a basic embodiment of the system, a user can demand his credential at will, upon providing proof of identity in the form of shared secret(s) that he has previously escrowed with the credential server. The shared secret may be chosen by the user, and could be easily remembered secrets such as: mother's maiden name, third grade teacher, etc. The user will respond to challenges from the server via a challenge-response protocol, with the server demanding correct answers to such questions prior to releasing the user's credentials. In another embodiment of the invention, a user's authentication credential can be stored on the server protected by a simple shared secret scheme such as a password, a biometric authentication scheme based on a fingerprint or retinal image, or a one-to-one hashed shared secret. In yet another embodiment of the invention, the user interacts with the server via a cryptographically camouflaged challenge-response protocol. In particular, if the user responds correctly to the server's challenges, the user will receive his authentication credentials. However, if the user responds incorrectly, such as might be the case with a hacker trying to break the system, the user will receive plausible and well-formed but invalid credentials. Furthermore, the authentication credential itself could be encrypted or camouflaged with an additional secret that is known only to the user. An authentication credential is said to be in cryptographically camouflaged form when it is embedded among many pieces of similar (pseudo-valid) data. These data are sufficiently different that the user can locate the correct piece without any difficulty, using a shared secret that he can remember. However, the pieces of data are also sufficiently alike that an intruder will find all of them equally plausible. Such a cryptographically camouflaged authentication credential can be provided to the user in either camouflaged or decamouflaged form that is, the decamouflaging can be performed at either the credential server or at the user's computer. The various embodiments of the invention described above provide one or more or the following advantages: No additional hardware is required for deployment. This is in contrast with hardware tokens such as smart cards where cards and card readers need to deployed in a widespread fashion.

Brief Summary Text (9):
(1) High user convenience. Roaming users need not carry tokens with them, but can demand them as required.

Brief Summary Text (10):
(2) Low administrative overhead. Users who have lost, misplaced or forgotten tokens do not require administrative intervention.

Detailed Description Text (8):
In either of the foregoing, the wallet could be installed either: 1) in the memory space of the software program, and/or subsequently 2) onto the hard drive or other physical memory of the computer. If only the former, the authentication credential would be destroyed when the session is ended. If the latter, the authentication credential could be available for use across multiple sessions on that particular computer. In either event, as the user roams to another computer, the process can be repeated to provide on-demand access to the needed authentication credential without the requirement of a physical token (even though the invention could also be used in conjunction with a physical token, as desired).

Detailed Description Text (10):
In one embodiment of the invention, the wallet might itself be protected by a shared secret. For example, FIG. 2 shows an exemplary embodiment of a wallet in which a private key is protected by a PIN. The PIN (more generally, a shared secret) might be the shared secret transmitted by the user to the Credential Server 160, as discussed previously, and the private key (more generally, the

authentication credential) in the wallet might be decrypted by Credential Server 160 and provided in the clear to the user at Browser 140. Alternatively, the entire wallet (including the authentication credential in encrypted form) might be provided to the user, for the user to decrypt locally at Browser 140. With either approach, the process of decrypting the PIN-protected authentication credential as follows. The user enters a PIN 200 (more generally, an access code) to unlock the wallet, and the PIN is passed through a one-to-one hash function 210. The hash function may also include a salt value or other security-enhancing feature, as will be appreciated by persons skilled in the art. The hashed value 215 of the entered PIN is compared with a stored hash value 220, which is the hashed value of the correct PIN. If the two hash values agree, the PIN is passed to decryption module 240. The private key which has been encrypted (with the correct PIN as the encryption key) and stored in field 230, is decrypted by decryption module 240, which is typically DES or some other cryptographic function such as, for example, triple-DES, IDEA or BLOWFISH. Hence, the decrypted private key 250 is released for use.

Detailed Description Text (14):
Referring now to FIG. 3, the authentication credential (e.g., private key) is protected via an access code as in FIG. 2. However, the one-to-one hash is replaced with a many-to-one hash, i.e., a hash in which many inputs produce (i.e., regenerate) the same hashed output. In an exemplary implementation, the many-to-one hash function 310 might hash six-digit codes to two-digit hash values. As in the conventional key wallet, the hashed value 315 of the entered PIN 300 is compared with the stored hash value 320, which is the hashed value of the correct PIN. If the two hash values agree, the key wallet opens. The private key is again stored encrypted in field 330 of the key wallet, with the correct PIN as the encryption key. When the correct PIN is entered, the stored encrypted key is decrypted and the correct private key 350 is released for use. However, since the hash function is many-to-one, there will be many different entered PINs that will satisfy the hash challenge to open the key wallet. (PINs that hash to the same hash value as the correct PIN, including the correct PIN, are referred to herein as pseudo-valid PINs.) For example, if the hash function hashes six-digit codes to two-digit hash values, there will be 10,000 six-digit pseudo-valid PINs that will open the key wallet, out of a total of 1,000,000 possible six-digit codes. Pseudo-valid PINs will all be passed to the decryption module 340 to decrypt the stored encrypted key to produce a candidate private key. However, all but one of these candidate private keys will be incorrect decryptions of the stored (correct) private key. Only when the entered PIN is the correct PIN will the correct private key be recovered.

Current US Cross Reference Classification (7):
705/65

Previous Doc Next Doc Go to Doc#

authentication credential) in the wallet might be decrypted by Credential Server 160 and provided in the clear to the user at Browser 140. Alternatively, the entire wallet (including the authentication credential in encrypted form) might be provided to the user, for the user to decrypt locally at Browser 140. With either approach, the process of decrypting the PIN-protected authentication credential as follows. The user enters a PIN 200 (more generally, an access code) to unlock the wallet, and the PIN is passed through a one-to-one hash function 210. The hash function may also include a salt value or other security-enhancing feature, as will be appreciated by persons skilled in the art. The hashed value 215 of the entered PIN is compared with a stored hash value 220, which is the hashed value of the correct PIN. If the two hash values agree, the PIN is passed to decryption module 240. The private key which has been encrypted (with the correct PIN as the encryption key) and stored in field 230, is decrypted by decryption module 240, which is typically DES or some other cryptographic function such as, for example, triple-DES, IDEA or BLOWFISH. Hence, the decrypted private key 250 is released for use.

Detailed Description Text (14):
Referring now to FIG. 3, the authentication credential (e.g., private key) is protected via an access code as in FIG. 2. However, the one-to-one hash is replaced with a many-to-one hash, i.e., a hash in which many inputs produce (i.e., regenerate) the same hashed output. In an exemplary implementation, the many-to-one hash function 310 might hash six-digit codes to two-digit hash values. As in the conventional key wallet, the hashed value 315 of the entered PIN 300 is compared with the stored hash value 320, which is the hashed value of the correct PIN. If the two hash values agree, the key wallet opens. The private key is again stored encrypted in field 330 of the key wallet, with the correct PIN as the encryption key. When the correct PIN is entered, the stored encrypted key is decrypted and the correct private key 350 is released for use. However, since the hash function is many-to-one, there will be many different entered PINs that will satisfy the hash challenge to open the key wallet. (PINs that hash to the same hash value as the correct PIN, including the correct PIN, are referred to herein as pseudo-valid PINs.) For example, if the hash function hashes six-digit codes to two-digit hash values, there will be 10,000 six-digit pseudo-valid PINs that will open the key wallet, out of a total of 1,000,000 possible six-digit codes. Pseudo-valid PINs will all be passed to the decryption module 340 to decrypt the stored encrypted key to produce a candidate private key. However, all but one of these candidate private keys will be incorrect decryptions of the stored (correct) private key. Only when the entered PIN is the correct PIN will the correct private key be recovered.

Current US Cross Reference Classification (7):
705/65

Previous Doc Next Doc Go to Doc#

L6: Entry 2 of 3 File: USPT Jul 17, 2001

DOCUMENT-IDENTIFIER: US 6263446 B1
**** See image for Certificate of Correction ****
TITLE: Method and apparatus for secure distribution of authentication credentials to roaming users

Application Filing Date (1):
19981119

Brief Summary Text (3):
To overcome the insecurity of the password, alternative technologies have been developed. One such technology is asymmetric key cryptography. In this technology, each user has two keys, a private key and a public key. The user performs a cryptographic operation (e.g., an encryption or a digital signature) on a digital quantity using his private key, such that the quantity may be authenticated by a verifier having access only to the user's public key. The private key therefore serves as the user's authentication credential. That is, the verifier need not know the user's private key in order to authenticate the user. Because the public key may be widely disseminated while the private key remains confidential, strong authentication is provided with enhanced security. Private keys are generally too long and complex for the user to memorize, and are therefore usually stored in software or hardware tokens, and interfaced with computers prior to use.

Brief Summary Text (4):
One such software token is the so-called software wallet, in which the private key is encrypted with a password or other access-controlled datum. In such software wallets, an intruder is not deterred from repeatedly trying passwords, in an exhaustive manner, until he recovers the private key. This poses analogous security risks to the simple password schemes described above. In addition, the software wallet is stored on a user's computer, which may be inconvenient if the user needs to freely roam from one location to another.

Brief Summary Text (5):
In contrast to software wallets, hardware tokens such as smart cards are more secure, and can be conveniently carried as the user roams. In a typical hardware smart card, the private key is stored in hardware, and protected by a watchdog chip that allows the user to access the private key, should he enter the correct password that unlocks the smart card. The smart card can even be configured so that, if a hacker attempts to guess passwords, the card locks up after a small number of successive missed attempts. The disadvantages of hardware token are: (1) roaming is restricted to locations where the appropriate token reader hardware is installed; (2) hardware tokens are expensive in contrast to software tokens; (3) hardware tokens must be physically carried wherever the user wishes to roam; and (4) hardware tokens are often lost, misplaced, or stolen.

Brief Summary Text (6):
Thus, while hardware token systems offer increased security, they have several disadvantages compared to software based systems. It would, therefore, be desirable to have a system that combines the best features of both hardware and software based systems.

Brief Summary Text (8):
The present invention discloses a method and apparatus for the on-demand delivery of authentication credentials to roaming users. Credentials are stored, delivered and transmitted in software, obviating the need for additional hardware. In a basic embodiment of the system, a user can demand his credential at will, upon providing proof of identity in the form of shared secret(s) that he has previously escrowed with the credential server. The shared secret may be chosen by the user, and could be easily remembered secrets such as: mother's maiden name, third grade teacher, etc. The user will respond to challenges from the server via a challenge-response protocol, with the server demanding correct answers to such questions prior to releasing the user's credentials. In another embodiment of the invention, a user's authentication credential can be stored on the server protected by a simple shared secret scheme such as a password, a biometric authentication scheme based on a fingerprint or retinal image, or a one-to-one hashed shared secret. In yet another embodiment of the invention, the user interacts with the server via a cryptographically camouflaged challenge-response protocol. In particular, if the user responds correctly to the server's challenges, the user will receive his authentication credentials. However, if the user responds incorrectly, such as might be the case with a hacker trying to break the system, the user will receive plausible and well-formed but invalid credentials. Furthermore, the authentication credential itself could be encrypted or camouflaged with an additional secret that is known only to the user. An authentication credential is said to be in cryptographically camouflaged form when it is embedded among many pieces of similar (pseudo-valid) data. These data are sufficiently different that the user can locate the correct piece without any difficulty, using a shared secret that he can remember. However, the pieces of data are also sufficiently alike that an intruder will find all of them equally plausible. Such a cryptographically camouflaged authentication credential can be provided to the user in either camouflaged or decamouflaged form that is, the decamouflaging can be performed at either the credential server or at the user's computer. The various embodiments of the invention described above provide one or more or the following advantages: No additional hardware is required for deployment. This is in contrast with hardware tokens such as smart cards where cards and card readers need to deployed in a widespread fashion.

Brief Summary Text (9):
(1) High user convenience. Roaming users need not carry tokens with them, but can demand them as required.

Brief Summary Text (10):
(2) Low administrative overhead. Users who have lost, misplaced or forgotten tokens do not require administrative intervention.

Detailed Description Text (8):
In either of the foregoing, the wallet could be installed either: 1) in the memory space of the software program, and/or subsequently 2) onto the hard drive or other physical memory of the computer. If only the former, the authentication credential would be destroyed when the session is ended. If the latter, the authentication credential could be available for use across multiple sessions on that particular computer. In either event, as the user roams to another computer, the process can be repeated to provide on-demand access to the needed authentication credential without the requirement of a physical token (even though the invention could also be used in conjunction with a physical token, as desired).

Detailed Description Text (10):
In one embodiment of the invention, the wallet might itself be protected by a shared secret. For example, FIG. 2 shows an exemplary embodiment of a wallet in which a private key is protected by a PIN. The PIN (more generally, a shared secret) might be the shared secret transmitted by the user to the Credential Server 160, as discussed previously, and the private key (more generally, the